MAHARISHI INTERNATIONAL UNIVERSITY

Computer Vision Facial Emotion Recognition on FER2013 Dataset

Authors:
Marjan Kamyab
Aisuluu Baktybekova
Bunyodjon Hoshimaliev
Ershad Islam
Mukhammadjon Yorkinov

Supervisor:
Anthony Sander

Final project

August 11, 2022

Acknowledgements

We would like to express our special thanks of gratitude to Prof. Anthony Sander for gave us the golden opportunity to do this wonderful project and for their patient guidance, enthusiastic encouragement, and valuable critiques during our Machine Learning class.

Abstract

Facial expression recognition (FER) plays a vital role in human-computer interaction and has become an essential field of choice for researchers in computer vision and artificial intelligence over the last two decades. As we know, an image's background or non-face areas will seriously affect the accuracy of expression recognition. In this project, we implement three different transfer learning strategies VGG16, DenseNet169, and MobileNet. Firstly we offer the Exploratory data analysis (EDA) to understand the dataset. Then we utilized transfer learning and feezing the feature extraction layers, which are not required for our datasets. We found that MobileNet and VGG16 achieved significant accuracy, and performance is deployed in a real-time framework that enables fast and accurate real-time output. As a result, superior performance to other state-of-the-art methods is achieved in facial expression databases FER2013.

Contents

Chapter 1

Introduction

Emotion recognition is one of the many facial recognition technologies that have developed and grown through the years. Currently, facial emotion recognition software is used to allow a certain program to examine and process the expressions on a human's face. Using advanced image dispensation, this software functions like a human brain that makes it capable of recognizing emotions too.

It is AI or "Artificial Intelligence" that detects and studies different facial expressions to use them with additional information presented to them. This is useful for a variety of purposes, including investigations and interviews, and allows authorities to detect the emotions of a person with just the use of technology.

The data was selected from FER-2013 [1] and it consists of 48x48 pixel gray-scale images of faces. The emotions on these faces are Angry, Disgust, Fear, Happy, Sad, Surprise and Neutral.

In this research, we applied 5 deep learning models. From these, the best 3 performed models were chosen and stated in this report in detail. These models were obtained by the help of VGG-16, DenseNet-169, MobileNet neural networks. At the end, these models were compared according to their accuracy and loss values.

[1] https://www.kaggle.com/datasets/msambare/fer2013

Chapter 2

Exploratory Data Analysis

2.1 Data set

We have selected the FER2013 dataset [1] for this experiment. The FER2013 is commonly used to train machine learning and computer vision algorithms. The data consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression into one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The training set consists of 22968 , 5741 in validation and 7178 images in test set. Figure 2.2 shows the 9 random images from dataset and Figure 2.1 shows data distribution in 7 class.

[1] https://www.kaggle.com/code/shawon10/facial-expression-detection-cnn/data

Figure 2.1: Data distribution of 7 classes

Figure 2.2: 9 random images from each emotion class

2.2 Image Data Generator

The dataset generated with the following parameters:

- horizontal_flip=True
- width_shift_range=0.1
- height_shift_range=0.05
- rescale = 1./255
- preprocessing_function=preprocess_fun
- color_mode = ”rgb”

In image processing, normalization is a process during which we change the range of each pixel, by reducing the value to the range from 0 to 1. We do this, because the computation of high numeric values can become very complex and computationally expensive. If we reduce the number of each pixel to fit between 0 and 1, the numbers will be small, and the computation becomes easier and faster. We achieve that by dividing all pixel values by the largest pixel value 255.

Chapter 3

Building Models

The models are implemented based on the TensorFlow deep learning API written in python language. The models are executed on the Windows operating system with 3.00 GHz processor and 8 GB RAM. This project was implemented based on the research paper; however, its updated with different hyperparameters and dataset [1].

3.1 VGG-16 Model

VGG16 is a convolutional neural network model proposed by K. Simonyan and A. Zisserman from the University of Oxford in the paper [2]. we have used VGG16 transfer model and updated the hyper parameters by utilizing common pattern by stacking Flatten layers, BatchNornalization, Dropout and the Dense as classification. Table 3.1 shows the hyper-parameter setting of the model. Figure 3.1 shows the model architecture details.

3.1.1 Training procedure and hyper-parameters setting

We have Functional, BatchNormalization, Flatten, Dense, Activation, Dropout for tuning the model. 22,968 observations were used for training and 7,178 used for testing. The modeling structure are as follows:

- Create a sequential model
- add base Model (VGG16)

  then Dropout with 0.5 value, flatten, and Batch Normalization utilized.
- add First, second and third block dense layer added with the following parameters:

  filters size 32, kernel_initializer='he_uniform', activation function ReLu, and dropout with 0.5 values.
- output layer: Finaly we applied the output layer with SoftMax function In addition, we used optim as optimizer, categorical crossentropy for loss value, and matrics as accuracy. Table 3.1 shows the hyperparameter Settings.

3.1.2 Result Analysis

The experiment result validates that the model performed well and achieved certain performance very quickly. Figure 3.2 shows Accuracy curve which increases as number of epochs increases and Figure 3.3 Loss curve which decreases gradually as number of epochs increases. At loss curve, we detect some anomaly at epoch 7. In addition, the models train and validation accuracy is not steadily stable; however, its still receive acceptable performance on the live prediction. The models' accuracy achieves its' maximum at 86% only in 14 epochs. On the other hand, loss value reaches its' minimum about 1.5 at epoch 15.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 vgg16 (Functional)          (None, 1, 1, 512)         14714688

 dropout (Dropout)           (None, 1, 1, 512)         0

 flatten (Flatten)           (None, 512)               0

 batch_normalization (BatchN  (None, 512)              2048
 ormalization)

 dense (Dense)               (None, 32)                16416

 batch_normalization_1 (Batc  (None, 32)               128
 hNormalization)

 activation (Activation)     (None, 32)                0

 dropout_1 (Dropout)         (None, 32)                0

 dense_1 (Dense)             (None, 32)                1056

 batch_normalization_2 (Batc  (None, 32)               128
 hNormalization)

 activation_1 (Activation)   (None, 32)                0

 dropout_2 (Dropout)         (None, 32)                0

 dense_2 (Dense)             (None, 32)                1056

 batch_normalization_3 (Batc  (None, 32)               128
 hNormalization)

 activation_2 (Activation)   (None, 32)                0

 dense_3 (Dense)             (None, 7)                 231

=================================================================
Total params: 14,735,879
Trainable params: 7,099,399
Non-trainable params: 7,636,480
```

Figure 3.1: Architecture details of VGG-16 model

Table 3.1: VGG-16 model hyper-parameters

Hyperparameter	Values
Number of epochs	[1-50]
Batch size	64
Input Shape	(48,48,3)
Dropout rate	0.5 (for all blocks)
Number of Dropout	3
Weights	imageNet
Kernel Initializer	he_uniform
Activation function type	ReLu, SoftMax
Filters size	32

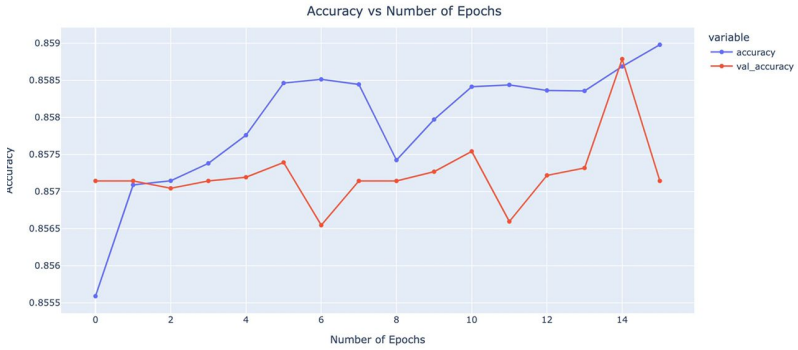

Figure 3.2: Accuracy curves of VGG-16 Model

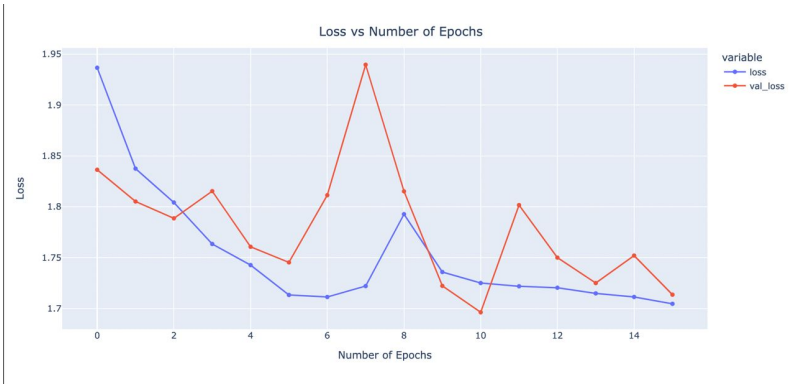

Figure 3.3: Loss curves of VGG-16 Model

3.2 DenseNet-169 Model

About DenseNet-169 and its' family.

DenseNet is a convolutional network where each layer obtains additional inputs from all preceding layers and passes on its own feature-maps to all subsequent layers; in other words, concatenation or "collective knowledge" from all preceding layers was used. DenseNets have several compelling advantages: they alleviate the vanishing-gradient problem, strengthen feature propagation, encourage feature reuse, and substantially reduce the number of parameters.
The DenseNet-169 model is one of the DenseNet group of models designed to perform image classification. The main difference with the DenseNet-121 model is the size and accuracy of the model. The densenet-169 is larger at just about 55MB in size vs the densenet-121 model's roughly 31MB size. Originally trained on Torch, the authors converted them into Caffe* format. All the DenseNet models have been pretrained on the ImageNet image database.

About our DenseNet-169 Model

In this classification model as Feature Extractor - DenseNet169 was used and as Classifier GlobalAveragePooling2D, stack of Dense layers with ReLu activation function, Dropout Layers and Dense layer with SoftMax was applied. Table 3.2 shows hyper-parameters, Figure 3.4 shows model architecture details.

3.2.1 Training procedure and hyper-parameters setting

The train and test data were same as previous 22,968 to 7,178 respectively. To tune the model GlobalAveragePooling2D, Dense and Dropout layers were used. The structure of the model was as follow:

- Feature detection
  We used DenseNet-169 for feature extraction purpose. The weight was provided by ImageNet and as input shape we provided (48,48,3) dimension.

- Transition and Classification Block
  We have used GlobalAveragePooling2D to pool in terms of average. Then applied Dense layer of size 128 with ReLu activation function and Kernel regulizer L2, dropout with rate of 0.3 to avoid overfitting. Later, we did same procedure twice but with 256, 512 filter sizes. As last, Dense layer was applied with size same as number of classes in this data and SoftMax was used as activation function to classify.

- Compilation of a Model and Model Fit
  To compile the model, we have used SGD optimizer for adjustment of weights and bias during Back Propagation, Categorical Crossentropy as a Loss function

because there were 7 categories present and Accuracy as metrics for evaluation. To fit the model, we split train data into 64 batch and trained it with 30 epochs. To not overfit and obtain the model with maximum accuracy, we assign callbacks as checkpoint and early stop. In early stop, the model obtained should have maximum accuracy after getting 5 unsuccessful epochs. In checkpoint, the model was serialized as HDF5 file depending on its maximum accuracy for further use.

3.2.2 Result Analysis

Figure 3.5, accuracy curve shows how training accuracy steadily and validation accuracy slightly fluctuatingly increases as epoch size increases and becomes less or more stable after 21 epochs. According to Figure 3.6, loss curve shows how train and validation loss decreases sharply as number of epochs increases and become stable after 4 epochs. As a result, this model performed with 63% of Accuracy and 1.05 Loss.

According to Figure 3.7, majority of test data were predicted correctly(diagonal part of the heatmap looks more dark colored). The most frequent emotion that was predicted correctly was Happy. The reason is Happy is the most frequent emotion. However there are some Error predictions. For example, disgust was correctly predicted zero times. This is because this emotion was the least frequent one. There was already some bias towards it from the beginning. Also, 233 of feared faces was predicted as sadness and 279 of sad faces were predicted as neutral. The same knowledge, we obtained from AUC scores of each emotion. According to Figure 3.8, the most accurate emotions are happy (97%) and surprise (96%). It was followed by Neutral(90%), disgust (89%) and anger (88%).

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 48, 48, 3)]       0

 densenet169 (Functional)    (None, 1, 1, 1664)        12642880

 global_average_pooling2d (G  (None, 1664)             0
 lobalAveragePooling2D)

 dense (Dense)               (None, 128)               213120

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 256)               33024

 dropout_1 (Dropout)         (None, 256)               0

 dense_2 (Dense)             (None, 512)               131584

 dropout_2 (Dropout)         (None, 512)               0

 classification (Dense)      (None, 7)                 3591

=================================================================
Total params: 13,024,199
Trainable params: 381,319
Non-trainable params: 12,642,880
```

Figure 3.4: Architecture details of DenseNet-169 model

Table 3.2: DenseNet-169 model hyper-parameters

Hyperparameter	**Values**
Number of epochs	[1-30]
Batch size	64
Input Shape	(48,48,3)
Dropout rate	0.3 (for 1st) and 0.5 (for all other two)
Number of Dropout	3
Weights	imageNet
Kernel Regularizer	L2(0.01)
Activation function type	ReLu, SoftMax
Optimizer	SGD
Filters size	128, 256, 512

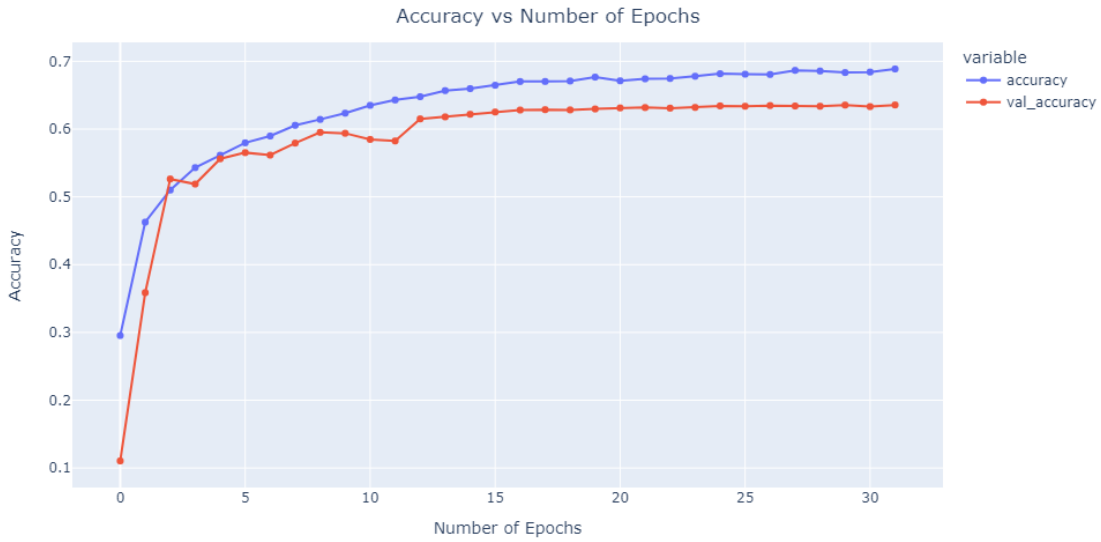

Figure 3.5: Accuracy curves of DenseNet-169 Model

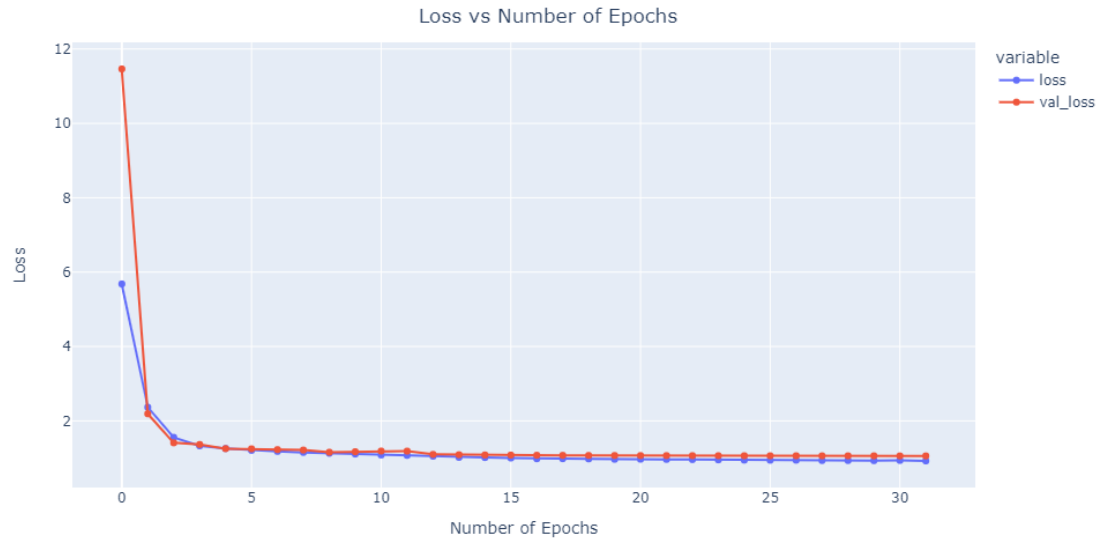

Figure 3.6: Loss curves of DenseNet-169 Model

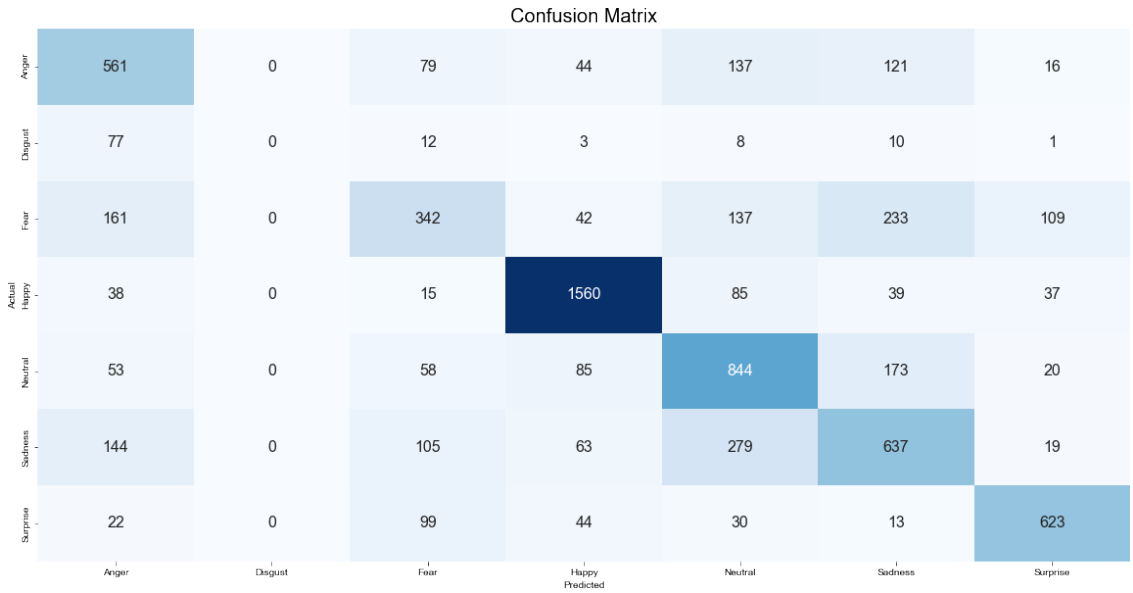

Figure 3.7: Heatmap of true vs predicted labels for DenseNet-169 Model

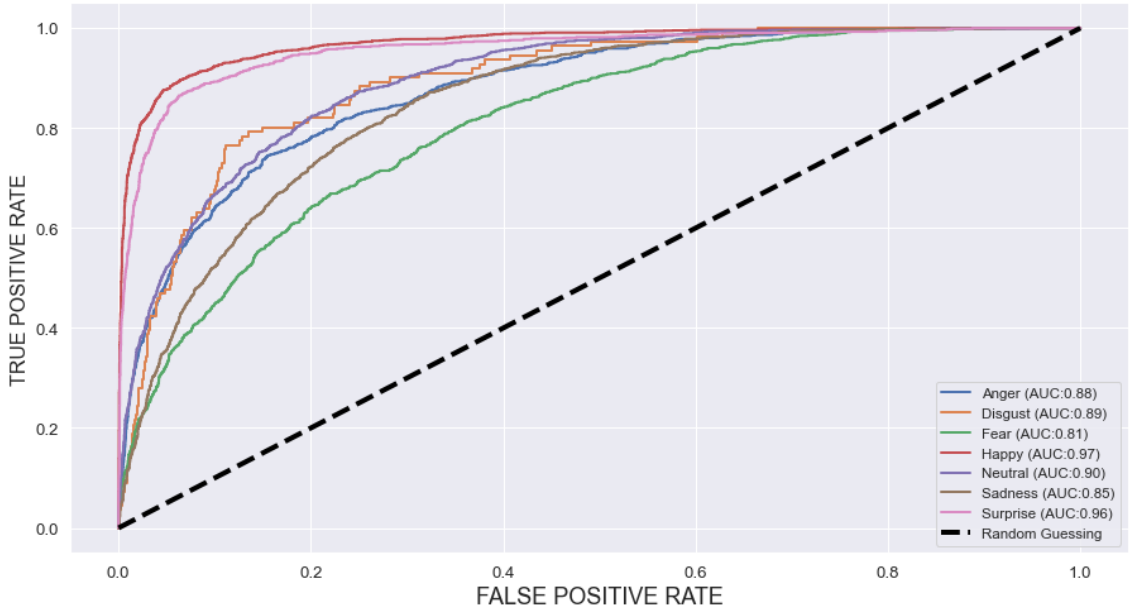

Figure 3.8: AUC scores for DenseNet-169 Model

3.3 MobileNet Model

The model is implemented based on the tensorflow keras deep learning API written in python language. The models are executed on the Windows operating system with 3.00 GHz processor and 8 GB RAM. We updated MobileNet model by using common pattern Dense layer,Batch Normalization, activation function. Table 3.3 shows the hyperparameter setting of the model and Figure 3.9 shows the model architecture details.

3.3.1 Training procedure and hyper-parameters setting

We have have update the MobileNet with Sequential Dense model, Flatten, Activation, Dropout for tuning the model. In training, 2968 datasets observations were used for training, 5741 in validation and 7178 images in test set. The modeling structure are as follows:

- Create a sequential model
- add base Model (MobileNet)

  then Dropout with 0.5 value, flatten, and BatchNormalization utilized.
- add First, second and third block with the following parameters:

  filters size 32, kernel_initializer='he_uniform', activation function ReLu, and dropout with 0.5 values.
- output layer: Finaly we applied the output layer with SoftMax function In addition, we used optim as optimizer, categorical crossentropy for loss value, and matrics as accuracy. Table 3.3 shows the hyperparameter Settings.

Layer (type)	Output Shape	Param #
mobilenet_1.00_224 (Functional)	(None, 1, 1, 1024)	3228864
dropout (Dropout)	(None, 1, 1, 1024)	0
flatten (Flatten)	(None, 1024)	0
batch_normalization (BatchNormalization)	(None, 1024)	4096
dense (Dense)	(None, 32)	32800
batch_normalization_1 (BatchNormalization)	(None, 32)	128
activation (Activation)	(None, 32)	0
dropout_1 (Dropout)	(None, 32)	0
dense_1 (Dense)	(None, 32)	1056
batch_normalization_2 (BatchNormalization)	(None, 32)	128
activation_1 (Activation)	(None, 32)	0
dropout_2 (Dropout)	(None, 32)	0
dense_2 (Dense)	(None, 32)	1056
batch_normalization_3 (BatchNormalization)	(None, 32)	128
activation_2 (Activation)	(None, 32)	0
dense_3 (Dense)	(None, 7)	231

Total params: 3,268,487
Trainable params: 3,244,359
Non-trainable params: 24,128

Figure 3.9: Architecture details of model using MobileNet

Table 3.3: Model hyper-parameters setting.

Hyperparameter	**Values**
Number of epochs	[1-100]
Batch size	64
Input Shape	(48,48,3)
Dropout rate	0.5 (for all blocks)
Number of Dropout	3
Number of Sequential Dense block	3
model type	Sequential
Activation function type	ReLu, SoftMax
Filters size	32

3.3.2 Result Analysis

The experiment result validates that the model performed well and achieved significant performance. Figure 3.10 shows the model performance Accuracy, AUC, Recall, and precision curve. The models train and validation AUC increases gradually and became stable after 10 epochs. The model performance in terms of accuracy, precision, recall are 0.9226, 0.8975 and 0.93 respectively . The models' accuracy were found as 96% whereas AUC is equal to 98% which is excellent result. Figure 2.3 This shows how the model improves loss values of train and validation performance over epochs.

According to Figure 3.12, majority of test data were predicted correctly(diagonal part of the heatmap looks more dark colored). The most frequent emotion that was predicted correctly was happy, surprise, fear , angry and so on. According to Figure 3.12, the most accurate emotions are happy (97%) and surprise (96%). It was followed by Angry and Fear (90%), sad and Disgust (92%) and neutral (88%). Table 3.4 average Accuracy, Precision, Recall, and F1-score of our model with MobileNet base model.

Figure 3.13 verify the prediction is true or not, by the new function we verify the percentage of true and false prediction of model on the test dataset. In this model, among the 36 images it predict 36 correct and zero incorrect.

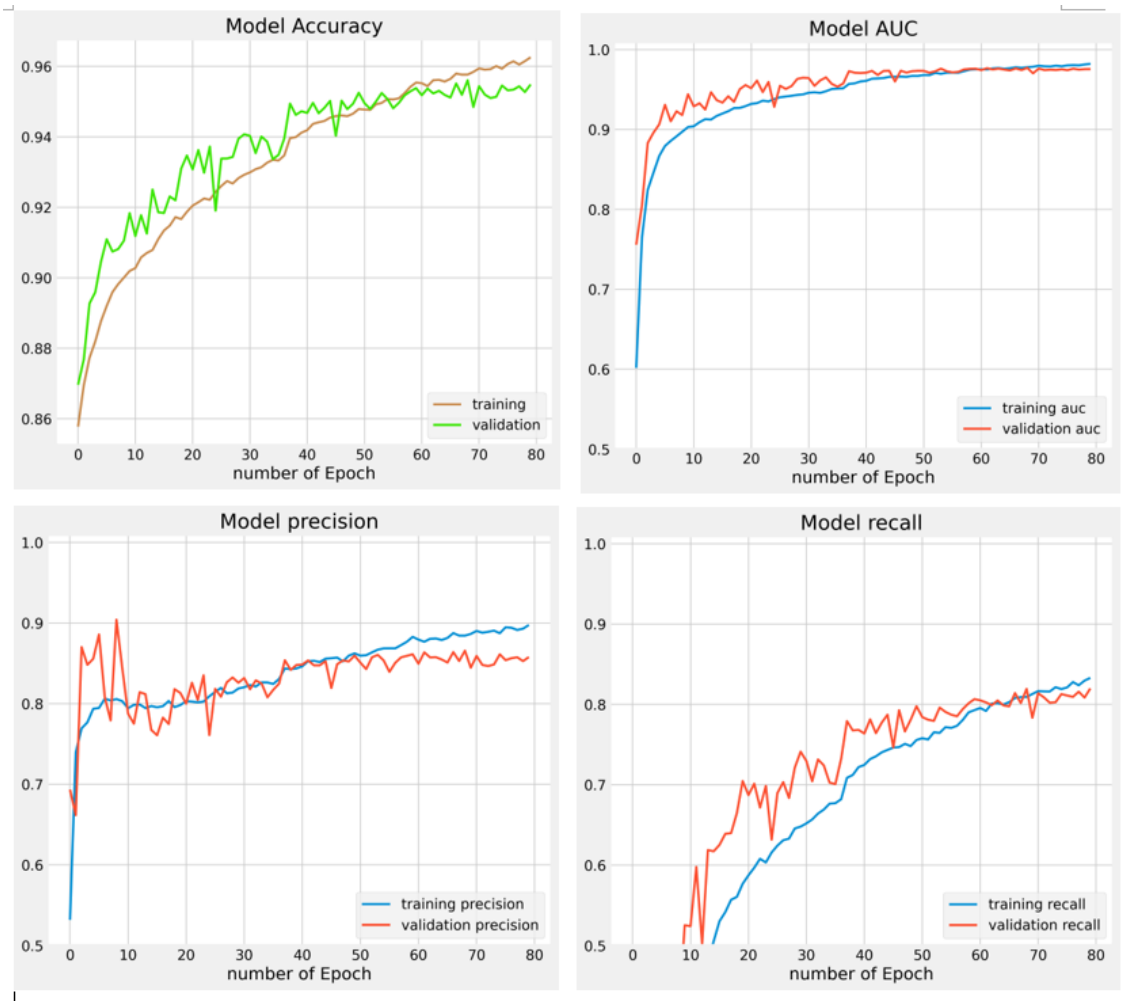

Figure 3.10: The figures show the model performance using the MobileNet network with our updated hyperparameters on the 80 epochs. Figure top (left) shows the accuracy of train and validation data, figure top (right) shows the AUC curve on train and validation data, figure bottom (right) shows the recall curve on train and validation data, figure bottom (left) shows the precision on train and validation data.

Figure 3.11: The figure show loss value improvement of model using the MobileNet network with our updated hyper parameters on the 80 epochs.

Table 3.4: Average Accuracy, Precision, Recall, and F1-score of our model with MobileNet base model.

Classes	precision	recall	f1-score
Angry	0.90	0.93	0.91
Disgust	0.94	0.92	0.93
Fear	0.88	0.93	0.90
Happy	0.97	0.97	0.97
Sad	0.92	0.92	0.92
Surprise	0.93	0.96	0.95
Neutral	0.96	0.88	0.92
accuracy			0.93
macro avg	0.93	0.93	0.93
weighted avg	0.93	0.93	0.93

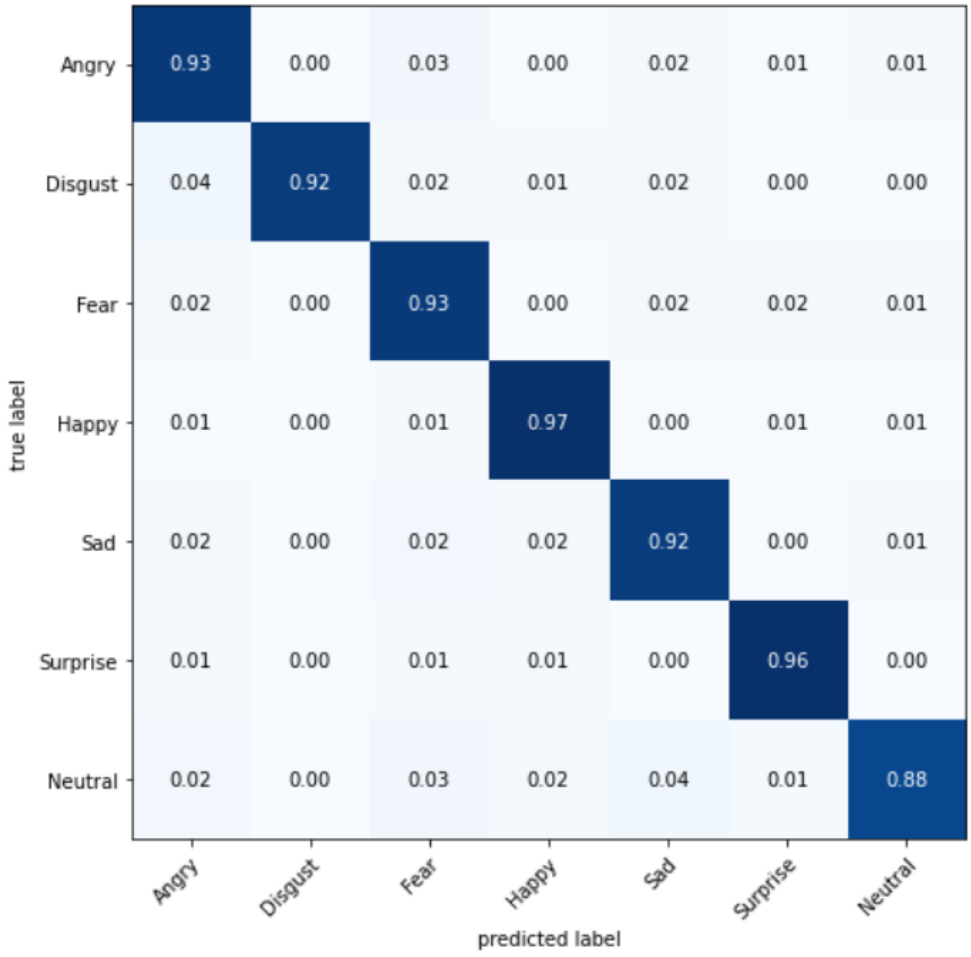

Figure 3.12: Heatmap of true vs predicted labels for model using MobileNet Model base model, e most accurate emotions are happy (97%) and surprise (96%). It was followed by Angry and Fear (90%), sad and Disgust (92%) and neutral (88%)

Figure 3.13: Plot of Actual vs Predicted results for Model 1 with percentage

3.3.3 Class Activation Maps (CAM)

Class Activation Maps (CAM) is a powerful technique used in Computer Vision for classification tasks. We have used Cam technique to verify our model performance. Figure 3.14 shows the 36 predicted images from different classes to understand which parts/pixels contributed more to the model's output.

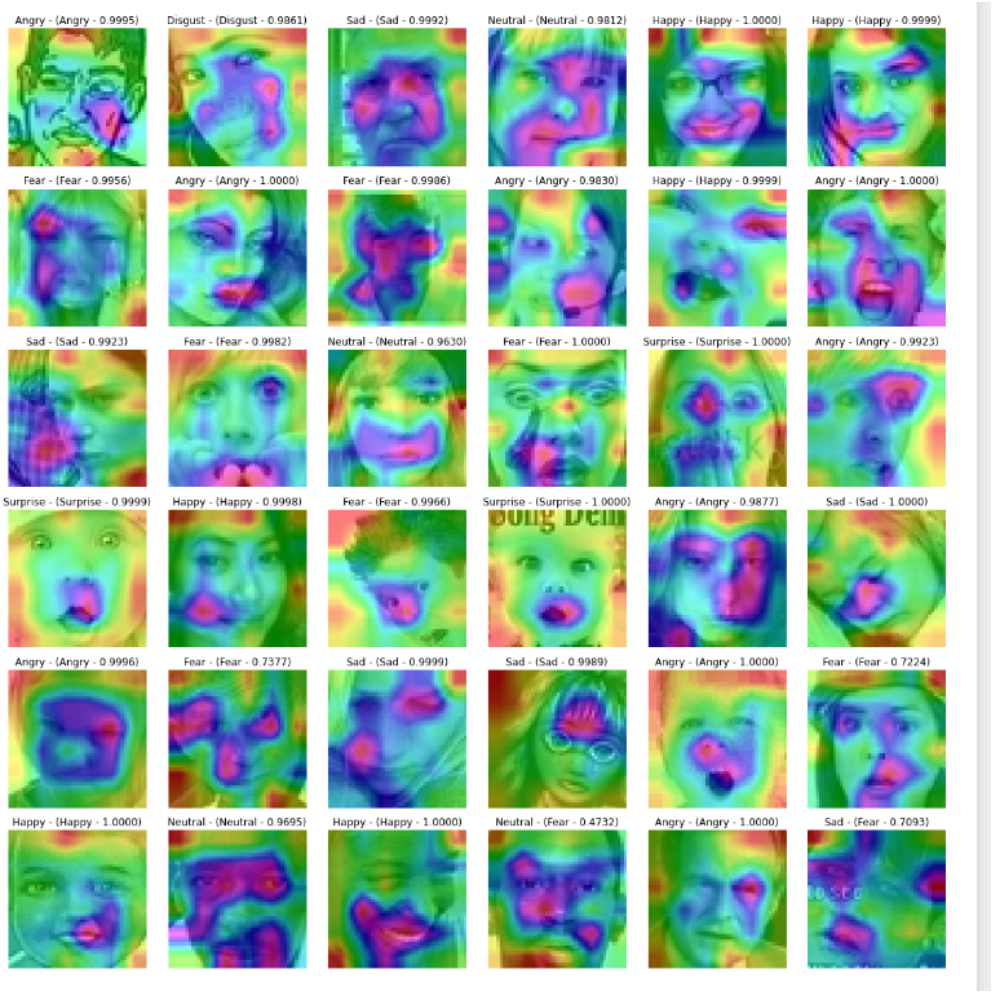

Figure 3.14: Cam technique to verify our model performance, darks color shows the parts/pixels contributed more to the output of the model

3.4 Model Comparison

In summary, the highest accuracy was obtained from VGG-16 Model, DenseNet-169 Model and MobileNet Model than other applied models. All had different approaches in terms of transfer learning techniques and the number of filters and dropouts. DenseNet-169 had a least accuracy of 63% among these 3 models. MobileNet had accuracy of 96% which is 11% higher compare to VGG-16. In terms of Loss value, again MobileNet was leader with 0.55 loss value. While DenseNet-169 had a loss of 1.05, VGG-16 had a value of 1.7. As a result, MobileNet performed extraordinary than other models in terms of both accuracy and loss value.

Chapter 4

Conclusion

This project presents three pre-trained convolutional neural network (DenseNet-169, VGG-16, MobileNet) architectures for classifying images in the FER2013 dataset. From the whole evaluation experiments, VGG-16 and DenseNet-169, and MobileNet showed their superior general performance over the other networks, while the VGG-16 and MobileNet had the best training and loss curves. MobileNet and DenseNet were more efficient and reasonable due to their fewer parameter numbers. VGG-16 had the lowest performance, but still, it is acceptable.

Bibliography

(1) Li, Q.; Yang, Y.; Guo, Y.; Li, W.; Liu, Y.; Liu, H.; Kang, Y. *IEEE Access* **2021**, *9*, 9318–9333.

(2) Simonyan, K.; Zisserman, A. *CoRR* **2014**, *abs/1409.1556*.

List of Figures

List of Tables